

# IMPLEMENTING SMP2 STANDARD WITHIN SIMTG SIMULATION INFRASTRUCTURE

*Claude CAZENAVE, William ARROUY*

*Astrium Satellites  
31 Rue des Cosmonautes  
31402 TOULOUSE*

[claude.cazenave@astrium.eads.net](mailto:claude.cazenave@astrium.eads.net)

[william.arrouy@astrium.eads.net](mailto:william.arrouy@astrium.eads.net)

## ABSTRACT

SimTG is the unique simulation infrastructure operationally deployed within all Astrium Satellites sites since 2009. It is composed of a set of generic software toolboxes including a simulation kernel (SimTG Kernel) and a Modelling Development Toolbox (SimMF). Since the very beginning, SimTG infrastructure has been developed upon the SMP2 standard. A full compliance with respect to SMP2 standard has been reached and demonstrated by model exchange with external simulation infrastructure.

The purpose of this paper is to present the following aspects:

- the standards selected and used to perform model exchanges
- the tooling used to develop models compatible with this standard
- the return on experience on this model exchange

## INTRODUCTION

Numerical simulation is a key element of satellite Functional Validation process in Astrium Satellites, starting from the use of Functional Validation Benches for the validation of AOCS algorithms up to Operation Simulators for the validation of spacecraft operations. As shown in Figure 1, various other simulation cases exist between these phases, the major ones being the Software Validation Facilities and the Real Time Test Benches used respectively for On Board Software validation and for integrated spacecraft checking. In the frame of CSO program, the French military observation spacecraft, successor of Helios, CNES is providing the operation simulator based on the models developed and validated at Astrium Satellites for the other simulation use cases described here above. While in previous programs, Astrium Satellites was used to deliver a complete simulator including the integrated models and the man machine interface to operate the simulator, it was the first time where we delivered standalone models not integrated into the simulation infrastructure (simulation kernel and man machine interface). As both simulation infrastructures, Basiles at CNES and SimTG at Astrium Satellites, are compatible with SMP2 standard, a natural choice was to rely on this standard to enable the porting of

the stand-alone models from one infrastructure to the other. However, as described in this paper, an additional standardisation effort, the so called “Reference Architecture”, has been required to ensure an efficient model exchange.

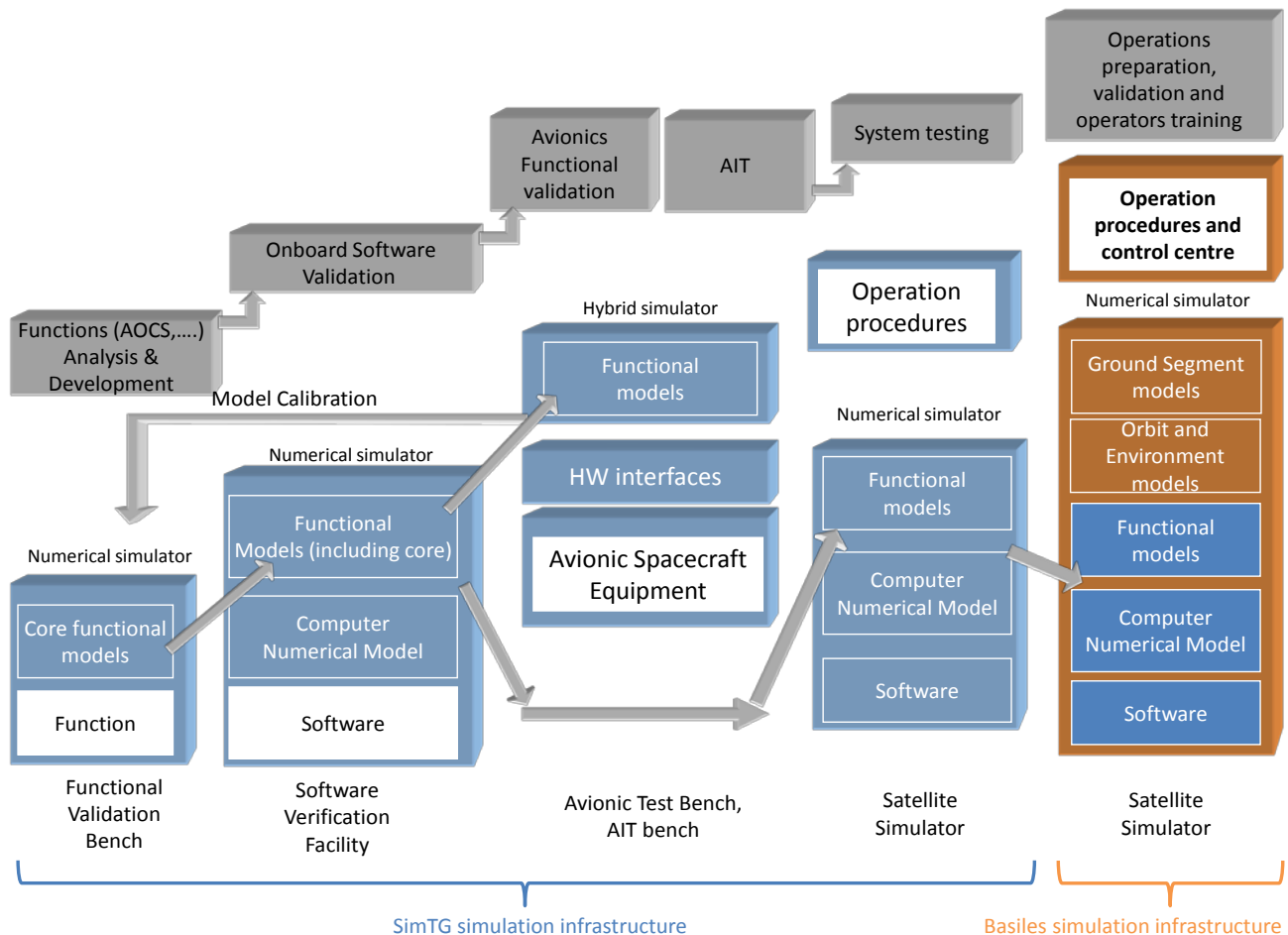


Fig1: Simulation life cycle diagram with Basiles additional use case

After the presentation of benefits brought by SMP2 standards, the next chapters will describe key elements of the selected Reference Architecture and modelling tool. Then, before concluding on the achieved results, the return on experience we got up to now in this process is described.

## BENEFITS OF STANDARDS

Simulation Portability Standard, SMP2 version 1.2 [1], was implemented in Astrium Satellites to share stand alone models between different simulation platforms. SMP2 standardization enables the exchange of models, however it does not guarantee that this exchange will be efficient. Indeed, SMP2 defines all what is required to ensure that a model can run on any simulation platform compatible with the standard, but there are limited generic software mechanisms, and no standard architecture to drive an efficient integration with other models. For instance, a SMP2-compatible model can provide a consumed power as an output using a SMP2 data field to be connected using a data flow, while other

models requesting these data can rely on other mechanisms like event or software interfaces and may also do not use the same conventions for units or the same dynamic for data exchange.

This is why it is well recognized by the community that there is a need to elaborate a so called Reference Architecture to define in a consistent way all the required data exchanges for the models to be integrated. Different Reference Architectures can be defined. The first one (REFA) was defined by ESOC[2] in a very pragmatic exercise in order to optimize the reuse of models currently available at ESOC. On the other hand, ESTEC defined the SSRA Reference Architecture taking into account industry needs. SSRA is not based on existing models. It is a standardisation approach mainly relying on de facto spacecraft design standards used to integrate equipment at electrical level. These electrical connections, are defining the system interfaces between the various on board equipment and this includes the communication protocols where applicable. In addition to these system interfaces, the other ones, called physical interfaces, are also standardised as much as possible to deal with remaining aspects like mechanical aspects (locations, forces ...), thermal aspects, etc.

Another Reference Architecture has been defined by CNES as part of the CNES ISIS initiative [4] for standardisation in spacecraft design. As SSRA, ISIS simulation model exchange standard relies on SMP2. It is based on a co-engineering activity made by CNES, Spacebel, Thales Alenia Space and Astrium Satellites. The same kinds of system and physical interfaces have been defined. The main concepts are similar, but the implementation and detailed definition of the interfaces differ from SSRA. Moreover, ISIS standard contains two different implementations of these interfaces to cope with existing industrial infrastructures. The first implementation is based on SMP2 event mechanisms and an evolution of SMP2 data flows, the second one is based on SMP2 software interfaces. These different implementations can be made compatible at software level thanks to the development of standard adapters. However, this illustrates the difficulty to make standardisation efforts compatible with heritages in Agencies and industry.

The next chapter describes how ISIS standard has been used to develop the CSO satellite and payload simulation models.

## **ISIS INTERFACES AND CONNECTION SERVICE**

Three different kinds of interfaces are described in the standard:

- System Interfaces: these are the electrical interfaces between equipment (e.g. a power line or a M1553 line)
- External interfaces: the interfaces to models external to spacecraft modelling (e.g. TM/TC links)
- Physical interfaces: the remaining interfaces which are dealing with physical interactions (e.g. inputs and outputs of orbit and environment model)

Astrum Satellites has recommended to use SMP2 software interfaces to implement all the interfaces where there is a asynchronous behaviour likely to be simulated. This is the case for all system interfaces and for external interfaces but not for physical interfaces where data flows may be used. The benefit of the software interfaces is twofold:

- It defines a non ambiguous signature of the exchanges thanks to method parameters that are verified at model compilation time
- It allows for the implementation of an asynchronous simulation when needed (i.e. an immediate computation of a connected model being called on its interface)

Following this approach, the use of data flows is limited to the models where the coupling is less critical in terms of dynamic of the exchange, like it is the case between an orbit model and all the models reading the orbital position. This also fits with existing models of that kind.

In addition to all these interfaces, a dedicated interface is defined to integrate differential equations (typically used for spacecraft dynamic computation). It is a dedicated service, called the “Central Solver”, provided by the simulation infrastructure. It has been decided to implement it using SMP2 software interfaces due to the accuracy of the interface signature and the need to ensure a precise sequencing of the computations.

Although it is possible to connect the various software interfaces using the SMP2 assembly file, Astrium Satellites has proposed to perform this integration thanks to a dedicated SMP2 service named “Connection Service”. This service brings a better decoupling of the equipment thanks to the introduction of an intermediate simulation model which represents an electrical line (i.e. a system interface). All the equipment simulation models and all the lines simulation models are created in a first step and then the Connection Service is activated (during SMP2 Connect call) to connect each equipment plug (or port) to the appropriate line. This way, all the connection information can be stored into a sub assembly that can vary according to the achieved integration status while keeping unchanged the sub-assemblies for the other models. The information required to do the connection is the name of the simulation model of the equipment and the name of the port where to connect the software interface for both ends of the line. This is in fact the name of the equipment and the name of the plug. An equipment simulation model do not know to which equipment it is connected but provides its name, the plug name and the type of the line to the Connection Service which returns the simulation model of the line. Figure 2 summarizes the approach with the use case of a power line.

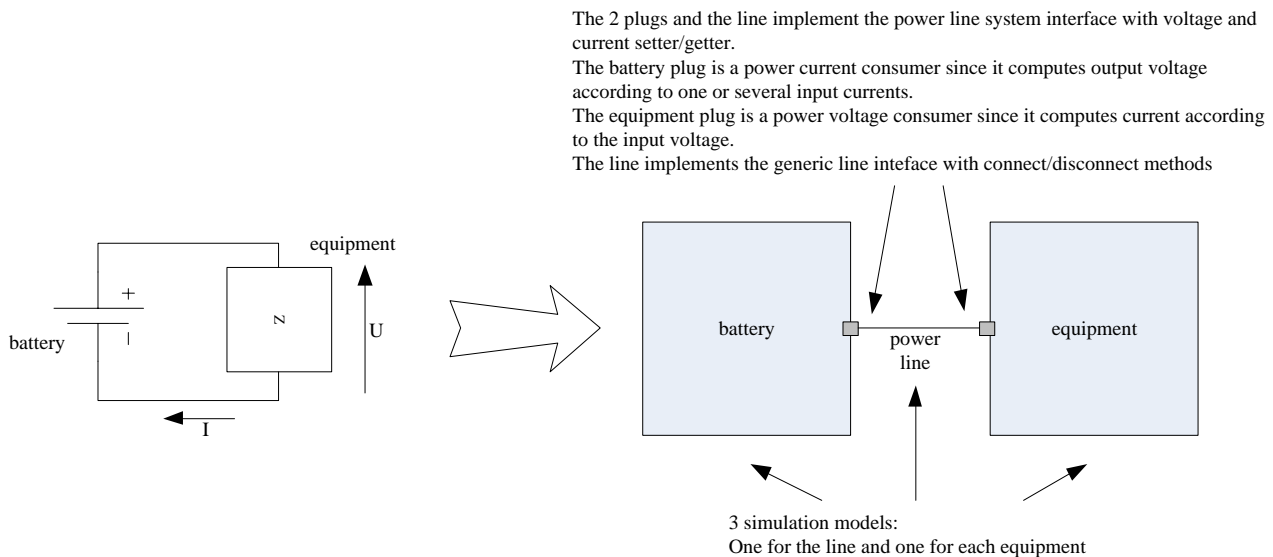


Fig2: System Interface concept with lines, plugs and Connection Service

In addition to this core role, the Connection Service provides many features like disconnecting the lines and monitoring of data exchanges. This approach is also very helpful to manage the simulation cases with hardware in the loop elements where the same assembly can be used and the connections established according to the availability of real equipment. Figures 3 and 4 present the software interfaces used for the Connection Service in the case of the power line system interface. The

Connection Service holds the various lines in dedicated SMP2 containers: the connection handlers. There is one connection handler for each type of line. For instance, *PowerLineHandler* (see Figure 3), is managing all the power lines.

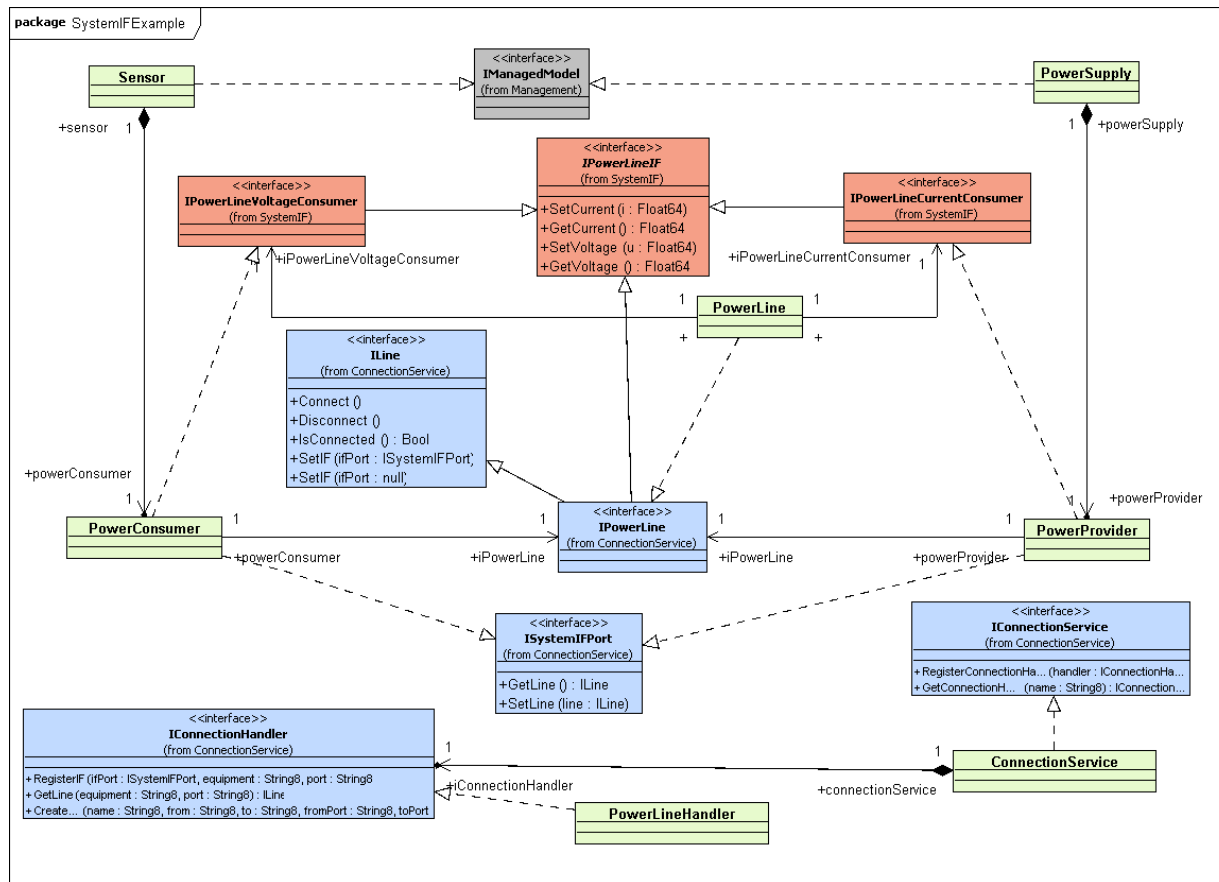


Fig3: Static diagram for Connection Service and power lines system interfaces

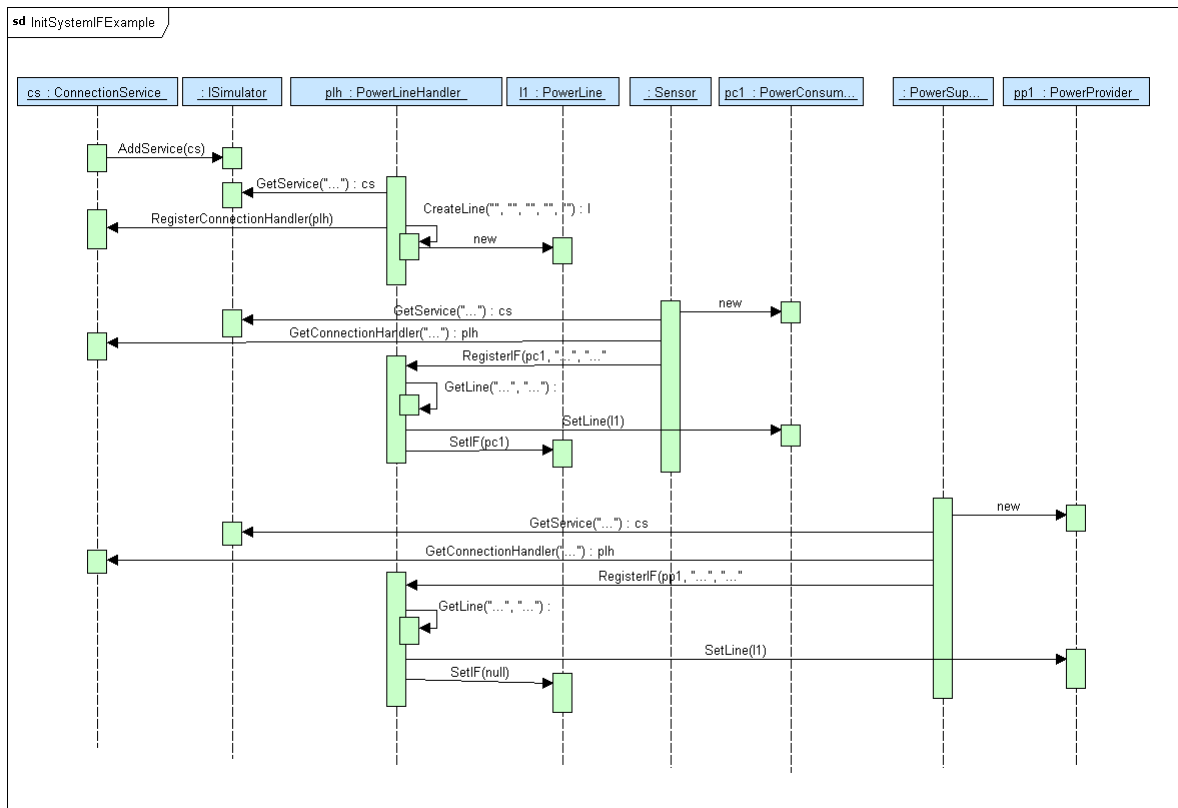


Fig4: Dynamic diagram for Connection Service and power lines system interfaces

While the previous diagrams only describe the power line case, the same approach is applicable to the other lines like High Power Command or M1553 lines.

In addition, the Connection Service is a SMP2-managed model where there exists a container for each type of line to be managed. The assembly file creates the Connection Service (as a model) and one sub model for each line in the appropriate container. Thanks to this approach the Connection Service can be integrated within any SMP2 compliant infrastructure.

The next chapter describes the use of this approach from a model developer point of view.

## USE OF SIMTG MODELLING TOOL: SIMMF

In order to develop simulation models which can be reused across the various simulation use cases and between spacecraft projects, Astrium Satellites has defined the SimTG modelling ICD [5]. This document contains the design of the models base classes to be compatible with SMP2 interface. The scope of this design is similar to what is achieved, as an example of implementation, by the Model Development Kit of SMP2 standard. In addition, this document describes how to breakdown a simulation model into a core part, introduced at FVB level, and complementary part used for the later use cases. Finally the document specifies the use of System Interfaces, Physical Interfaces and External Interfaces in a fully consistent way with SSRA and ISIS. Each ISIS System Interface, defined at C++ level, is integrated in SimTG modelling ICD to ensure the compatibility. The next figure provides an overview of the simulation model breakdown and the use of the various interfaces.

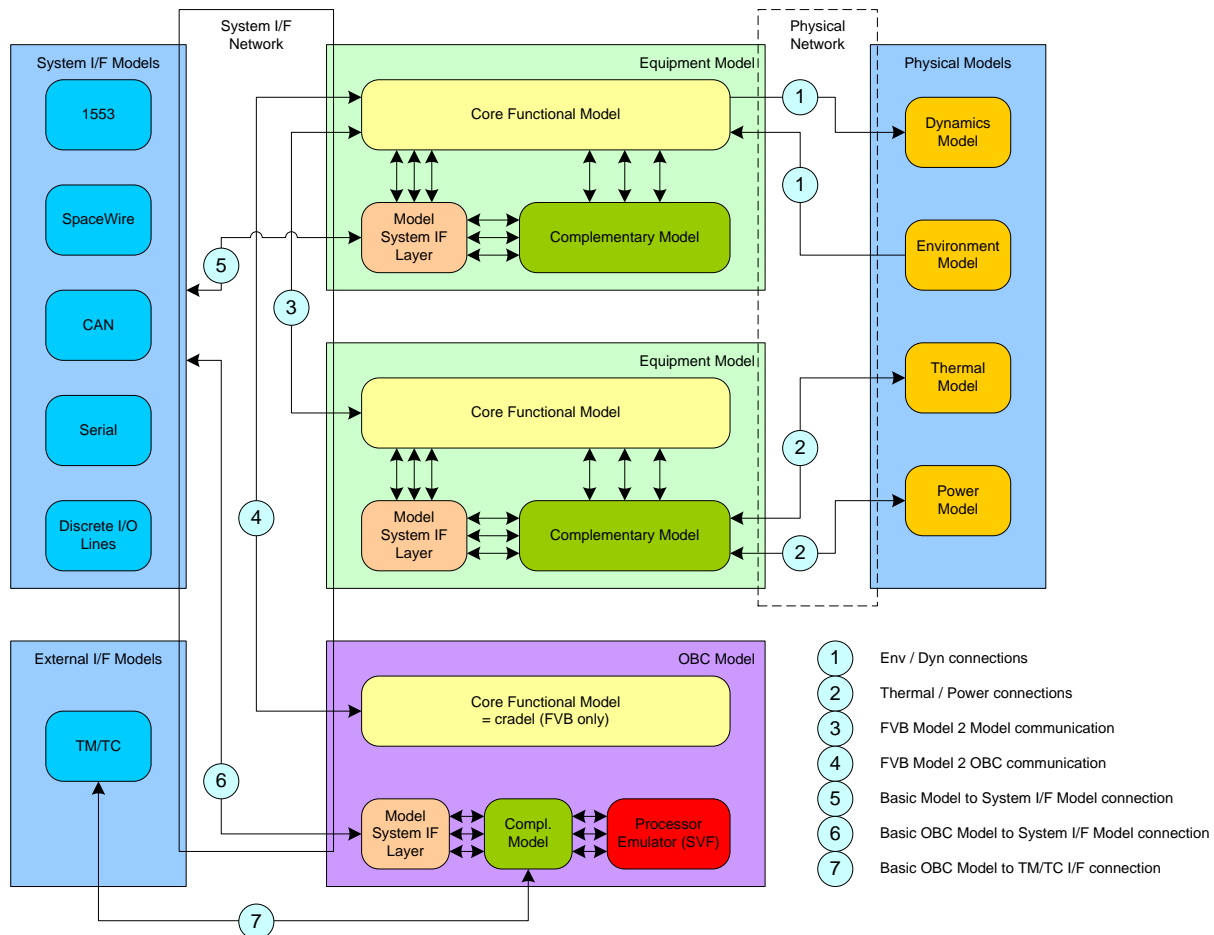


Fig5: SimTG modelling ICD models breakdown and interfaces

As part of SimTG infrastructure, a modelling tool has been developed named as SimMF [6]. The tool is used to develop and to integrate all the simulation models. Each functional model is composed of a core part, used for the Functional Validation Bench, and a complementary part, used for the other simulation use cases. Integration in a single tool of all the means to develop simulation models is a key requirement of SimMF. This includes model specification, documentation, interface definition, C++ coding (with consistency checks), compiling, integration and testing. As an example, the next figure show the definition of a GNSS model made of core and complementary sub models.

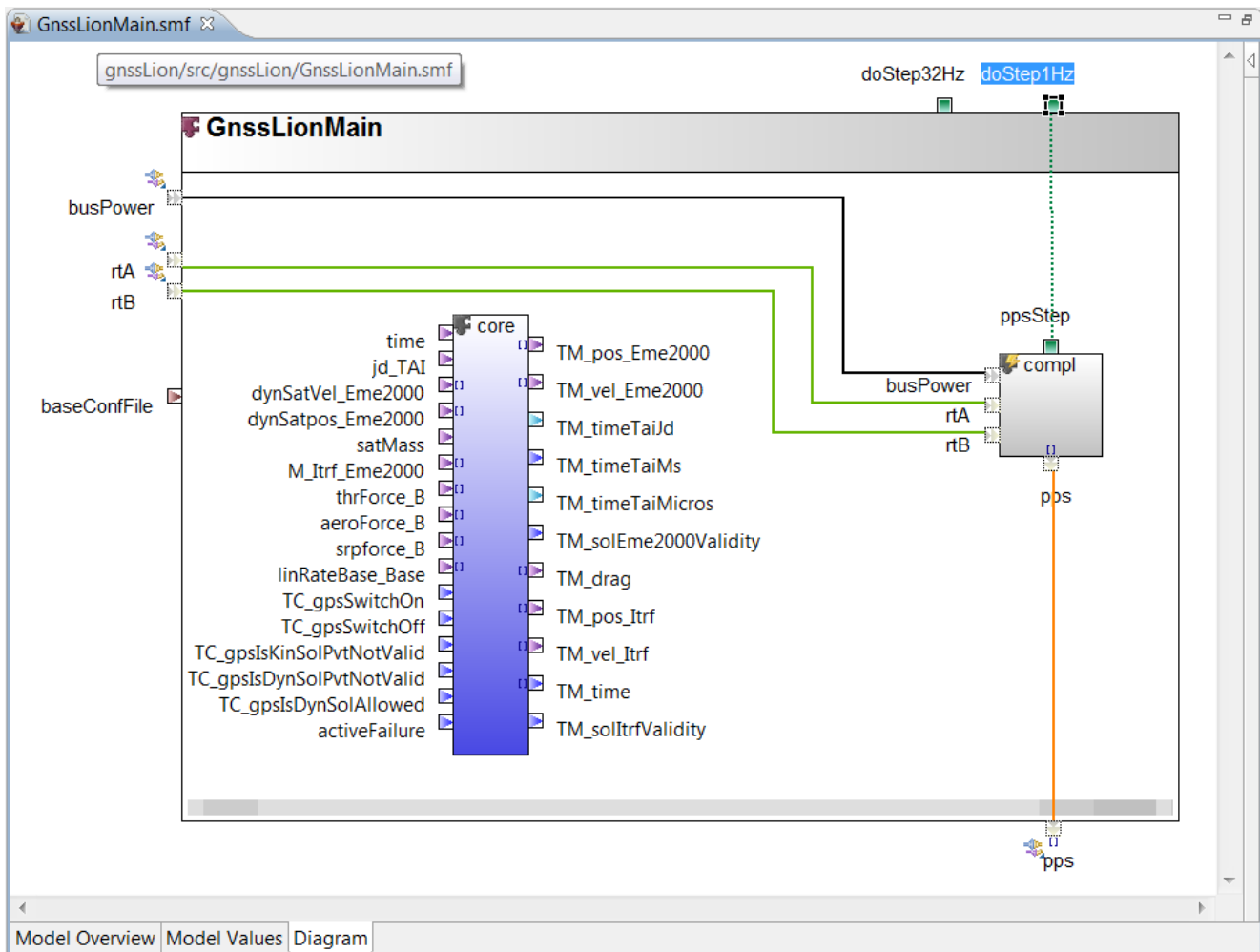


Fig6: A GNSS model made of a core part, a complementary part and system interfaces.

In the here above example, the top level model owns ports to be connected with one power line, two M1553 remote terminal busses and a synchronisation line. These ports are not really part of the top level model since the modelling is performed inside the complementary model. Thus SimMF allows defining proxy port (as it is the case here above) to route the connections to sub-models. This feature is very helpful for dealing with an arbitrary level of decomposition of the models (usually up to 5 or 6 levels for complex models). In addition, two ports respectively named `doStep32Hz` and `doStep1Hz` are defined to schedule the model. These are SMP2 entry points that can be scheduled using a SMP2 schedule file.

The next diagram shows the design of the GNSS complementary model. It shows an example of the targeted level of complexity and justifies the use of a single methodology for the design and integration of models and sub-models. This kind of model, in charge of electrical modelling, is asynchronous. All data exchanges, when data flows are used, are performed automatically without any scheduler interaction. When the value of a model output, connected to another model input(s), is changing, the connected model(s) will automatically perform its computations as soon as the first model has finished its ones. All this data propagation, asynchronous and conditional, is achieved thanks to SimMF code generation and SimTG kernel [7] data features.





- Serial line : for serial interfaces; one for the receive and one for the transmit
- Sync line : for synchronisation pulses
- HPC line : for high power commands
- M1553 line : for a M1553 remote terminal or a bus controller

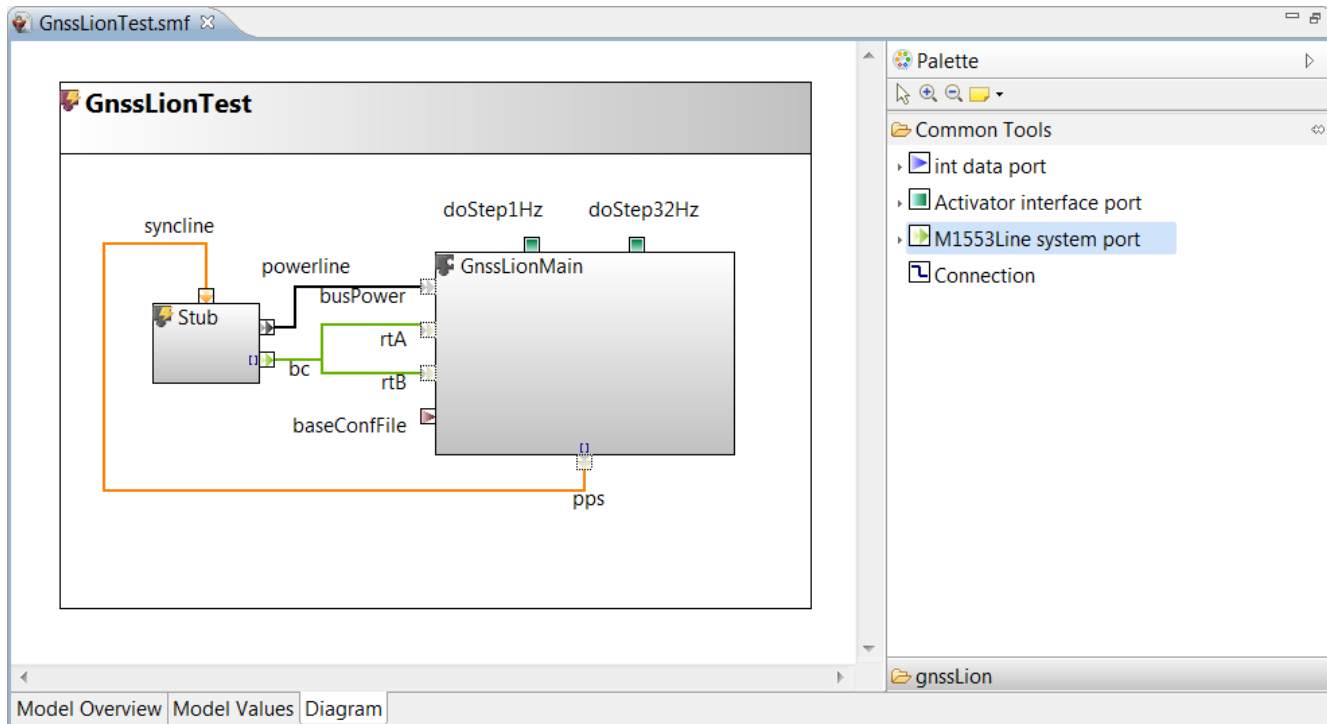
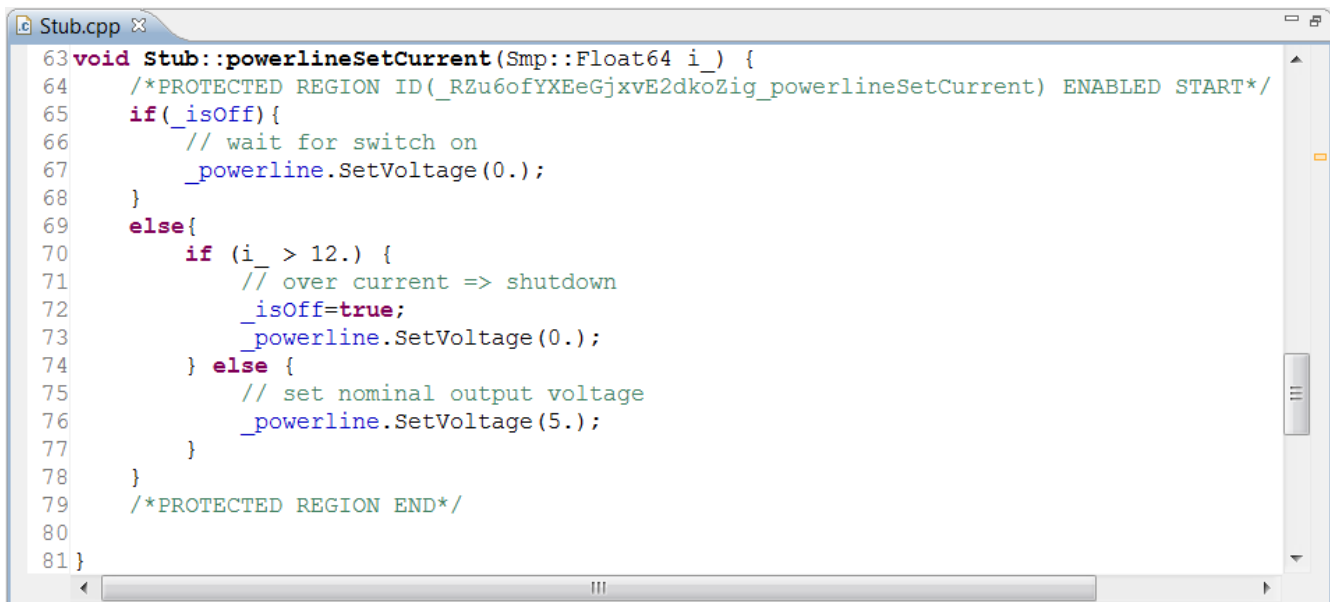


Fig8: integration test case for the GNSS model with a stub

As one can see, this stub owns a power line port. It is the power provider; it provides voltage to the connected equipment and consumes current required by the equipment. To be notified on any change on the current value on the line, a C++ method is automatically generated by the tool. The method name is composed of the name of the port and the name of the power line system interface relevant method(s). In that case only the SetCurrent() method is required and the user has to fill the C++ code in the protected area as described in the next snapshot, nothing else is required to deal with the power line.



```
63 void Stub::powerlineSetCurrent(Smp::Float64 i_) {
64     /*PROTECTED REGION ID(_RZu6ofYXEeGjxvE2dkoZig_powerlineSetCurrent) ENABLED START*/
65     if(!_isOff){
66         // wait for switch on
67         _powerline.SetVoltage(0.);
68     }
69     else{
70         if (i_ > 12.) {
71             // over current => shutdown
72             _isOff=true;
73             _powerline.SetVoltage(0.);
74         } else {
75             // set nominal output voltage
76             _powerline.SetVoltage(5.);
77         }
78     }
79     /*PROTECTED REGION END*/
80 }
81 }
```

Fig10: C++ code sample for a power line current consumer

In addition to the C++ code generated automatically for each model, SimMF can generate SMP2 catalogue and assembly files. Usually there exists a top level diagram with an instance of a simulation model for each equipment. One can generate either a C++ model of this diagram (all the integration is performed using compiled C++ code) or a set of SMP2 assembly files to create all the instances (all the integration is performed at run time by the simulation infrastructure). While the first approach is used at Astrium Satellites using the SimTG kernel simulation infrastructure, the second approach is used for testing the simulation models delivered to CNES on Basiles infrastructure.

Using this tooling, many SMP2-based stand alone models have already been developed, validated and delivered to CNES. The next chapter provides our return on experience versus the applied standards and the simulation infrastructures.

## RETURN ON EXPERIENCE

### With Regard To SMP2 Standard

The publication of data made of arrays of simple data types or structures is working well. This is not the case for strings of variable length. This kind of data is very useful and shall be supported in an efficient way. For the time being, the work-around solution consists in using a fixed length character array but the mapping of the C++ standard library strings is required. This has to be performed thanks to a change on SMP2 standard or at least in the incoming ECCS standard (ECCS E40 07).

Access to models data fields is not always performed in the same manner since the standard provides two different possibilities: use of C++ pointers or use of C++ methods for getter and setter. There is a discrepancy between the implementation on SIMSAT, the ESOC simulation infrastructure, and the one on Basiles (code generation for getters and setters) and it is not clear how to ensure that getters and setters are always used in place of direct access through pointers. This prevents the encapsulation of simple data types into C++ object, at least in a portable way, that are required for automatic data

propagation or a generic failure injection service. This aspect has to be enhanced in the incoming ECSS standard.

The instantiation of a SMP2 model using an assembly file is well defined. However, this is not defined for instantiation of SMP2 services. The current work-around is to implement services as simulation models that automatically register themselves as an SMP2 service. It would be better to describe how to instantiate SMP2 services in the standard.

Using the integration approach described here above, only the top level models are integrated using SMP2 mechanisms. The lower level models, even if their content is fully visible from an SMP2 infrastructure, are integrated using dedicated means, at C++ level, like conditional propagation for instance. Astrium Satellites considers that it is a valid approach for model exchange for the following reasons:

- it sounds difficult to reuse or exchange sub parts of complementary models
- the integration of the complementary model and the core model has to rely on C++ direct accesses or mechanism that are not part of SMP2 standard (for instance conditional propagation)
- the integration at runtime of all the models and their sub-models including all the information exchanges between all the sub-models do not seem to be efficient for both initialization time and runtime performance

Since this use case is not described in the standard, Astrium Satellites has found a work-around solution that consists in publishing the sub models without creating new instances when requested by the model factory (the sub-models are all instantiated when the top level model is created). For the sake of clarification, Astrium Satellites is requesting to describe a standard solution for this use case.

Using SimTG kernel scheduler, the models can post events with parameters. These are only input parameters but this feature is often used for modelling. In addition, it is also very helpful, for simulator users, to interact with the simulation. This feature is not available in SMP2, nor in the incoming ECSS standard, and it implies to find case by case work-arounds to ensure compatibility (for instance to define several entry points, one for each value of the parameter). Thus, such a change would be very beneficial.

### **With Regard To ISIS Standard**

Both the system interfaces and the Connection Service defined in [4] have been used without any changes. It is the result of a formalized approach based on software interfaces part of the document and directly copied into C++ headers that are used to generate the simulation infrastructure and the models.

### **With Regard To Simulation infrastructures: Basiles and SimTG**

The two integration methods described in the previous chapter (C++ or SMP2 assembly) gives the same simulation results, as expected. However, some fixes were required on both Basiles and SimTG infrastructure to manage all the SMP2 use cases that were not enough validated before.

Compared to SimTG, Basiles does not provide a generic failure injection service. This service is very helpful for simulation operators but also for models validation. In order to perform the validation of the models running on Basiles, an SMP2 service, providing the same feature as the ones available on SimTG kernel [7], has been developed. It allows for the translation of validation procedures from SimTG kernel into Basiles environment. This service is very generic and is likely to be helpful as part of the standard.

## CONCLUSION

Stand-alone models, part of the CSO platform, have been delivered to CNES. This includes the OBC model, the power, all the sensors and all the actuators. For each of these models, tests running on SimTG environment have been translated for compatibility with Basiles. All these tests have to be rerun successfully on Basiles.

The model exchange is very efficient since there is no adaptation to be performed on the models. However, it is not the case for the validation. The reason is that both simulation environments use different test languages (TCL for Basiles and Java for SimTG) and that, in addition, the simulation infrastructures do not provide identical services to interact with the simulation models. Thus, there is a need to progress in the area of models testing and this should be part of next R&D activities.

Finally, all the problems described in the return on experience at SMP2 level, even if there exists some workarounds, should be taken into account to improve the quality of the standard.

## REFERENCES

- [1] SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099, Issue 1.2, 28-Oct-2005
- [2] Spacecraft Simulator Reference Architecture Specification, REFA, Software Requirements Specification Volume 1 (EGOS-SIM-REFA-SRS-1001), Issue 1.2, 12-Dec-2008
- [3] The Space Simulation Reference Architecture – Reference Architecture Specification Volumes 1-3, (SSRA.REP.001, SSRA.REP.002, SSRA.REP.003), Issue 1.0, 27-May-2010
- [4] ISIS Training Operation and Maintenance Interface Specification (ISIS-SIM-IF-305-CNES), Issue 4.0, 9-Sept-2011
- [5] SimTG Modelling ICD (SimTG-ICD-0001-ASTR), Issue 2.0,
- [6] SimTG Model Development User Manual (SIMTG-UM-0009-ASTR), Issue 1.0, 23-Jan-2012
- [7] SimTG Kernel User Manual (SIMTG\_UM-0008-ASTR), Issue 2.0, 20-Jan-2012